John von Neumann Institute for Computing

Parallel Computing with Low-Cost FPGAs: A Framework for COPACOBANA

Tim Güneysu, Christof Paar, Jan Pelzl, Gerd Pfeiffer,
Manfred Schimmler, Christian Schleiffer

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr,
F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 741-748, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

http://www.fz-juelich.de/nic-series/volume38

Parallel Computing with Low-Cost FPGAs: A Framework for COPACOBANA

Tim Güneysu[1], Christof Paar[1], Jan Pelzl[3], Gerd Pfeiffer[2], Manfred Schimmler[2], and Christian Schleiffer[3]

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
E-mail: {gueneysu, cpaar}@crypto.rub.de

[2] Institute of Computer Science and Applied Mathematics, Faculty of Engineering
Christian-Albrechts-University of Kiel, Germany
E-mail: {gp, masch}@informatik.uni-kiel.de

[3] escrypt GmbH - Embedded Security, Bochum, Germany
E-mail: {jpelzl, cschleiffer}@escrypt.com

In many disciplines such as applied sciences or computer science, computationally challenging problems demand for extraordinary computing power, mostly provided by super computers or clusters of conventional desktop CPUs. During the last decades, several flavours of super computers have evolved, most of which are suitable for a specific type of problem. In general, dedicated clusters and super computers suffer from their extremely high cost per computation and are, due to the lack of cheap alternatives, currently the only possible solution to computational hard problems. More recently, emerging low-cost FPGAs tend to be a very cost-effective alternative to conventional CPUs for solving at least some of the computational hard problems such as those appearing in cryptanalysis and bio-informatics.

In cryptanalysis, breaking symmetric or asymmetric ciphers is computationally extremely demanding. Since the security parameters (in particular the key length) of almost all practical crypto algorithms are chosen such that attacks with conventional computers are computationally infeasible, the only promising way to tackle existing ciphers (assuming no mathematical breakthrough) is to build special-purpose hardware. Dedicating those machines to the task of cryptanalysis holds the promise of a dramatically improved cost-performance ratio so that breaking of commercial ciphers comes within reach.

This contribution presents the realization of a very generic framework for the COPACOBANA (Cost-Optimized Parallel Code Breaker) machine[1]. COPACOBANA consists of up to 120 low-cost FPGAs and can be realized for US$ 10,000 while being able to outperform conventional computers by several orders in magnitude, depending on the application. The presented framework allows for a simple distribution of parallelizable tasks and corresponding control mechanisms. With the framework at hand, the overhead and, as a consequence, the time to deploy custom made applications on the COPACOBANA platform is minimized. Exemplarily, we show how cryptanalytical applications can be based on top of the framework, resulting in a very low cost per computation.

1 Introduction

Although modern computers provide high performance at low cost, the cost-performance ratio of special purpose hardware can be better by several orders of magnitude, dependent on the problem. The main drawback of special purpose hardware, the high development costs and the lack of flexibility is overcome by the advent of modern reconfigurable hardware, i.e. Field Programmable Gate Arrays (FPGA). FPGAs are not able to reach the

performance of Application Specific Integrated Circuits (ASIC) but they provide the possibility of changing the hardware design easily while outpacing software implementations on general purpose processors.

A common approach to increase the performance of a single unit, i.e., a personal computer, is to build a cluster of low-cost off-the-shelf machines. One weakness of this solution is the low communication bandwidth, since most clusters are loosely coupled based on slow external peripherals introducing a high communication overhead. Another drawback of such a general-purpose architecture is its cost: solving a specific problem usually does not involve all features of the underlying hardware, i.e., parts of the hardware are running idle since they are not used at all. Another approach is to construct a parallel computer where several processors share the same system resources. This allows for very fast data throughput but requires system design constraints to remain scalable. In general, the same principle works for special purpose hardware as well.

In contrast to software development, it is a tedious task to develop hardware. This has changed with powerful software tools for design and simulation combined with the high performance of today's computers. The design of a single chip circuit becomes sufficiently affordable and reliable for small groups of engineers. The demands on design complexity are exponentially increasing when a system of several chips has to be set up since the communication between the chips gets more sophisticated due to asynchronous interfaces.

This paper presents a low-cost approach of a highly parallel FPGA cluster design with intended but not limited use in cryptography. COPACOBANA (Cost-Optimized Parallel Code Breaker) utilizes up to 120 Xilinx Spartan-3 FPGAs connected through a parallel backplane and interfaces the outside world through a dedicated controller FPGA with an Ethernet interface and a MicroBlaze soft-processor core running uClinux. The final production cost of the system was planned not to exceed material costs of US$10,000.

In the following we will give a brief overview about the hardware structure of the system, then we will provide details on the framework and programming of the system followed by currently implemented applications, applications under development and speed estimations for the system.

2 Architectural Overview

COPACOBANA consists of three basic blocks: one controller module, up to twenty FPGA modules and a backplane providing the connectivity between the controller and the FPGA modules, see Fig. 1.

The decision to pick a contemporary low-cost FPGA for the design, the Xilinx Spartan3-1000 FPGA (XC3S1000, speed grade -4, FT256 packaging) was derived by an evaluation of size and cost over several FPGA series and types. This Spartan3 FPGA comes with 1 million system gates, 17280 equivalent logic cells, 1920 Configurable Logic Blocks (CLBs) equivalent to 7680 slices, 120 Kbit Distributed RAM (DRAM), 432 Kbit Block RAM (BRAM), and 4 digital clock managers (DCMs)[2].

A step towards an extendable and simple architecture has been accomplished by the design of small pluggable FPGA modules. We decided to settle with small modules in the standard DIMM format, comprising 6 Xilinx XC3S1000 FPGAs. The FPGAs are directly connected to a common 64-bit data bus on board of the FPGA module which is interfaced to the backplane data bus via transceivers with 3-state outputs. While disconnected from

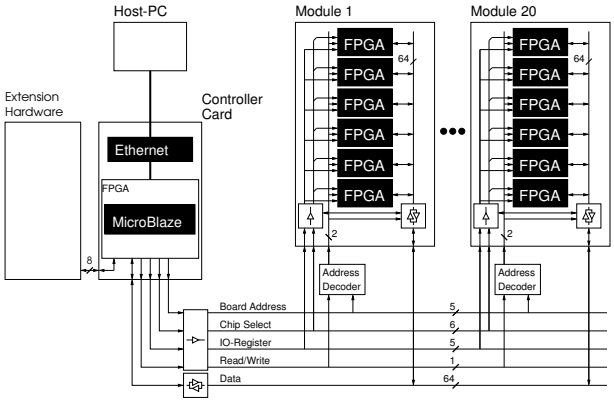

Figure 1. Architecture of COPACOBANA

the bus, the FPGAs can communicate locally via the internal 64-bit bus on the DIMM module. The DIMM format allows for a very compact component layout, which is important to closely connect the modules by a bus. From the experience with current implementations on the same FPGA type, we dispense with active cooling of the FPGAs using 80 mm fans. Depending on the heat dissipation of future applications, additional heat sinks and head spreaders might be options for an upgrade.

For the controller board we decided to use a commercially available solution to reduce the costs. In terms of flexibility we went for an FPGA board being large enough to host a CPU core and some additional logic for the bus access to move time critical and recurring bandwidth intensive tasks into hardware. We finally settled for a board with another Spartan-3S1000 FPGA as well as RAM, Flash, two serial ports and a 100 MBit/s Ethernet interface. We synthesized a Xilinx MicroBlaze system-on-chip running uClinux to which the backplane and communication logic is connected via a dedicated coprocessor interface.

The software stack includes uClinux because of its multitasking and networking abilities, i.e. access to the COPACOBANA machine is granted via a proprietary text-based protocol over TCP/IP networks. The application suite for the controller implements all necessary commands for configuring the slave FPGAs, communicating with the slave FPGAs and even for detecting how many FPGAs are actually present in the machine.

The backplane hosts all FPGA-modules and the controller card. All modules are connected by a 64-bit data bus and a 16-bit address bus. This single master bus is easy to control because no arbiter is required. Interrupt handling is totally avoided in order to keep the design as simple as possible. If the communication scheduling of an application is unknown in advance, the bus master will need to poll the FPGAs.

Moreover, the power supply is routed to every FPGA module and the controller interface. The backplane distributes two clock signals from the controller card to the slots. Every FPGA module is assigned a unique hardware address, which is accomplished by Generic Array Logic (GAL) attached to every DIMM socket. Hence, all FPGA cores can have the same configuration and all FPGA modules can have the same layout. They can easily be replaced in case of a defect. Figure 3 (Appendix) shows a complete COPACOBANA system. The backplane bus is calculated to run at a maximum frequency of

70 MHz, however currently it is working at 25 MHz due to power dissipation.

The authors are currently not aware of any similar project, realizing a FPGA cluster using low-cost components. However, there are several super computing architectures like the BEE2[3] or the commercial Computing Node Platform by Syntective Labs[4], both focusing on high computational power as well as on sophisticated node interconnections, introducing high costs per chip and computation.

3 Framework Structure

Along with the construction of the hardware, we developed a framework[5] for the ease of developing applications. It includes libraries for the software of the host computer, the hardware and software set for the controller, and a small stub residing in every slave FPGA. The framework provides a transparent communication from the host computer to the slave FPGAs.

The communication endpoint in every slave FPGA is a memory block, accessible from the FPGA implementation and the host computer as well as a status register providing several information about the current state of operation of the FPGA, i.e. a progress indicator and several status bits. Finally, the framework stub of the slave FPGAs includes arbitration and bus access mechanisms for the backplane as well as clock and reset generation for the actual computing application.

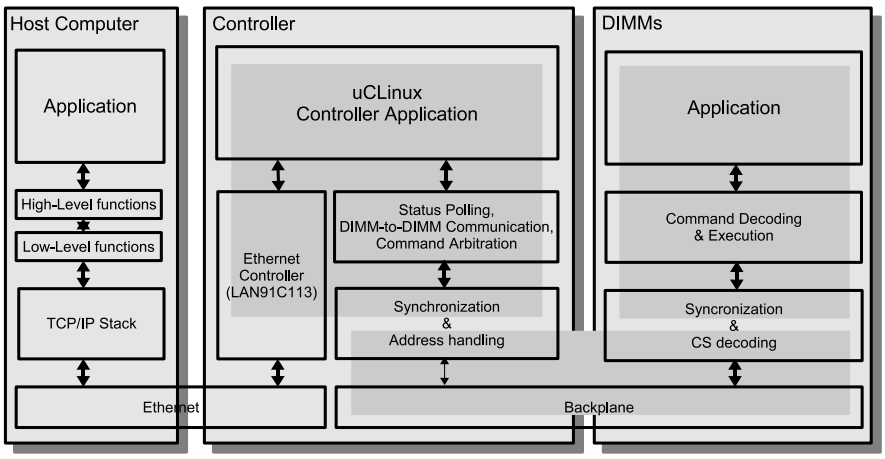

Figure 2. Framework structure; independent clock domains are marked by shadows

The controller implements the bus mastering and address generation on the lowest bus-synchronous layer. All higher layers are synchronous to the MicroBlaze CPU but do not have to be synchronous to the backplane. The hardware is able to perform read and write operations on the shared memories of the slave FPGAs by a single software command. Furthermore, it uses time slots without commands from the CPU to continuously poll the status registers of the slave FPGAs caching them in the controller for faster access. An additional benefit is the automatic detection of FPGAs that lock up and halt. Finally, the

polling system can emulate interrupts, i.e., it notifies the CPU upon detecting a service-request flag in the status register, and it is prepared to emulate a multi-master bus by moving data from one slave to another upon detection of a communication-request flag in the status register. The highest operational layer of the controller is a software component providing a textual command interpreter through a network socket.

3.1 COPACOBANA Host-PC Library

On the host computer, a single library was created to operate the COPACOBANA system. It implements a class establishing and monitoring a connection to the hardware system through a network connection to the command interpreter mentioned before. The network connection is handled transparently for the user, i.e., the library can even auto-detect COPACOBANA systems within the broadcast domain of the current network. The library is implemented in GNU-C++ and therefore is portable to a large number of operating systems and hardware platforms.

Besides the Ethernet port, the controller provides two serial ports, where one of them transmits a serial console from the uClinux operating system, which is not required in normal operation but can be useful for debugging purposes. The second serial port can be connected to an optional graphic display board, to show progress data and system information. The display board can be accessed through the COPACOBANA host libraries.

3.2 Configuration

To lower the system complexity, the same addressing and data infrastructure is used both for communication and for configuration of the FPGAs, i.e. the lowest eight bits of the data bus are connected to the slave parallel configuration ports of all slave FPGAs. The address decoding logic is used for selecting the FPGAs to be configured. With this powerful system it is possible to configure a single FPGA, a set of FPGAs or all of the 120 FPGAs at the same speed.

To allow for this approach we did not connect the feedback lines of the configuration ports to the controller, i.e., the configuration speed must stay below the maximum allowed speed of the slave FPGAs (50 MHz in this case). The configuration concept was described in detail by Pfeiffer et al.[6]

4 Applications

Though the design was intended for use in cryptanalytic applications, it is not necessarily restricted to it. In the following we will present a few applications already implemented and planned to be run on COPACOBANA as well as a brief overview about the special properties of applications to unveil the power of COPACOBANA since the compromise of low-cost hardware implies several limitations.

4.1 Cryptanalytical Applications

The first realized proof-of-concept application running on COPACOBANA was an exhaustive key search of the Data Encryption Standard (DES)[1]. The algorithm has a key size of

56 bit and a block size of 64 bit. The current implementation is capable of doing a key recovery in approximately 6.4 days at a total key test rate of 65.28 billion keys per second. Each FPGA is running four fully pipelined DES engines running at 136 MHz.
The latest presentation of a special purpose system for a key search called Deep Crack was presented by the Electronic Frontier Foundation in 1998 and consisted of 1856 ASICs that were able to do a key search in 56 hours. The overall system cost was US$250,000. Since then, no other DES cracker became publicly known.

The COPACOBANA is not limited to do only a stand-alone exhaustive key search. In case that only partial information of a plain-ciphertext for the exhaustive key search is available, the COPACOBANA can return potential key candidates to the host computer so that only a small subset of remaining keys needs to be checked in a second software based step. For example, encryption schemes relying on an effective key length of 48 bits due to export regulations are still in use. Having only 28 bits of known plaintext, a key search application running at 100 MHz can return a key candidate after 2^{28} encryptions on average. In a second processing step, these remaining 2^{20} key candidates can easily be tested in software using a conventional PC in reasonable time.

Besides the straightforward brute force attacks, another field of application of the COPACOBANA is to support cryptanalysis for public-key algorithms. The most efficient generic attack on Elliptic Curve Cryptosystems (ECC) is a parallel implementation of the Pollard-Rho algorithm capable to solve the Elliptic Curve Discrete Logarithm Problem (ECDLP) in approx. $\sqrt{\pi b/2}/\left|\mathcal{P}\right|$ steps where b is the bitlength of the underlying prime field and $\mathcal{P}$ the number of processors used. Hence, the ECDLP for small bitlengths, e.g., $b = 80$ and $b = 112$ bits standardized by the Standards for Efficient Cryptography Group (SECG)[7,8] can be solved using a set of COPACOBANAs within two weeks and several months, respectively[9].

Other applications under development are co-factorization algorithms to support the sieving step of large integer factorization based on the General Number Field Sieve (GNFS), e.g., to tackle the security of RSA-768[10]. For this purpose the Elliptic Curve Method factoring mid-sized integers can be implemented on a COPACOBANA allowing for high-performance sieving.

4.2 Further Applications

As mentioned before, COPACOBANA is not necessarily restricted to cryptanalytical applications. Basically every application with low memory demands and low to medium communication requirements can be run on the system with high performance, provided that the FPGA implementation is efficient. One candidate of an algorithm to be evaluated soon is the Smith-Waterman[11,12] algorithm for local sequence alignment, e.g. for determining similar regions between protein sequences.

5 Speed Estimations

The current version of the framework is still in early beta stage. However, with the current design we can make reasonable assumptions on throughput and latency. The layered design behaves like a pipeline, i.e. the latency for single operations is a little higher but bulk data transfers can be done at a reasonable speed. Writing a single data block of 64 bits from the

controller CPU to a slave takes approximately eight clock cycles whereas a read operation takes twelve clock cycles. Assuming that every data transfer on the bus can be done in 4 clock cycles (a conservative value since especially block-write transfers are much faster), the raw data bandwidth of the bus is approximately 6.25 GBit/s at 25 MHz.
During configuration of the slave FPGAs, a more moderate timing is used to guarantee reliability, i.e. an FPGA configuration takes about 1.5 s. Both, the controller and the 100 MBit/s Ethernet interface are a major bottleneck in the communication to the host computer, i.e. the payload data rate is limited to about 30 MBit/s.

6 Conclusion and Future Work

The work at hand presents the design and first implementation of a generic framework for the cost-optimized parallel FPGA cluster COPACOBANA. COPACOBANA can be built for less than US$ 10,000 and hosts 120 low-cost FPGAs which can be adopted to any suitable task which is parallelizable and has low requirements for communication and memory.

We implemented and described a complete framework using only freely available development tools such as uClinux, allowing for the rapid development of applications without deep knowledge and understanding of the architecture of the system. The use of the framework reduces overhead and realization time of custom made applications for COPACOBANA to a minimum, making COPACOBANA an extremely interesting alternative to expensive super computers or clusters of conventional PCs.

COPACOBANA is the first and currently only available cost-efficient design to solve cryptanalytical challenges. COPACOBANA was intended to, but is not necessarily restricted to solving problems related to cryptanalysis. Almost certainly, there will exist more interesting problems apart from cryptology which can be solved efficiently with the design at hand. In an ongoing project, we plan to apply the Smith-Waterman algorithm for scanning sequences of DNA or RNA against databases.

Currently, we are also evaluating an upgrade of the controller interface to Gigabit Ethernet and to change the processor or the FPGA type at the same time, to enhance data throughput.

References

1. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, *Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker*, (2006). `http://www.copacobana.org/paper/copacobana_CHES2006.pdf`
2. Xilinx, Spartan-3 FPGA Family: Complete Data Sheet, DS099, (2005). `http://www.xilinx.com`
3. C. Chang, J. Wawrzynek and R. W. Brodersen, *BEE2: A high-end reconfigurable computing system*, IEEE Design & Test of Computers, pp. 114–125, (2005).
4. Syntective Labs, *Pure computational power – the computing node platform – CNP*, (2007). `http://www.synective.com/synective_products.pdf`
5. C. Schleiffer, *Hardware and software implementation of a communication layer for a parallel FPGA cluster with application in cryptography*, Master's thesis, Horst Görtz Institute, Ruhr University of Bochum, (2007).

6. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer and M. Schimmler, *A configuration concept for a massively parallel FPGA architecture*, (2006).
7. Certicom research, *Standards for efficient cryptography — SEC 1: elliptic curve cryptography*, Version 1.0, (2000).
Available at `http://www.secg.org/secg_docs.htm`.
8. Certicom research, *Standards for efficient cryptography — SEC 1: recommended elliptic curve domain parameters*, Version 1.0 (2000).
Available at `http://www.secg.org/secg_docs.htm`.
9. T. E. Güneysu, *Efficient hardware architectures for solving the discrete logarithm problem on elliptic curves*, Master's thesis, Horst Görtz Institute, Ruhr University of Bochum, (2006).
10. M. Šimka, J. Pelzl, T. Kleinjung, J. Franke, C. Priplata, C. Stahlke, M. Drutarovský, V. Fischer and C. Paar, *Hardware factorization based on elliptic curve method*, in: IEEE Symposium on Field-Programmable Custom Computing Machines - FCCM 2005, Napa, California, USA, J. Arnold and K. L. Pocek, Eds., pp. 107–116, (2005).
11. C. W. Yu, K. H. Kwong, K. H. Lee and P. H. W. Leong, *A Smith-Waterman systolic cell*, in: Proc. 13th International Workshop on Field Programmable Logic and Applications — FPL 2003, pp. 375–384, (Springer. 2003).
12. G. Pfeiffer, H. Kreft and M. Schimmler, *Hardware enhanced biosequence alignment*, in: International Conference on METMBS, pp. 11–17, (CSREA Press, 2005).

Appendix

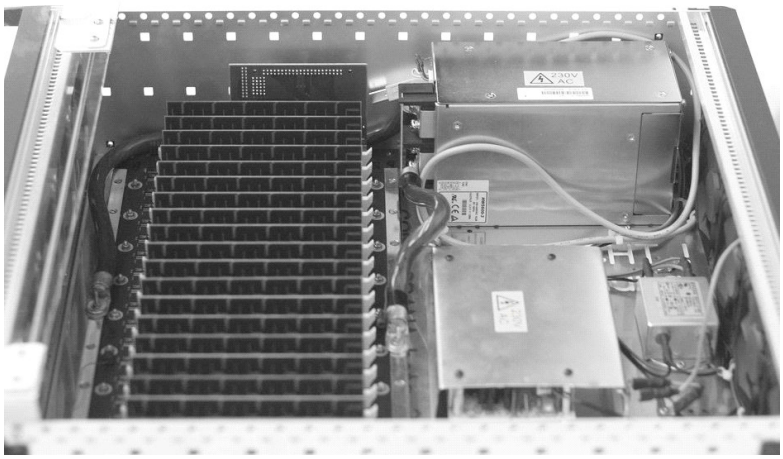

Figure 3. Complete COPACOBANA system with 20 DIMM modules